Assignment 3

USE CASE: Elevator Operation

Primary Actor(s): Passenger

Stakeholders and Interests:

Passenger: Wants to efficiently and safely use the elevator to reach their desired floor.

Building Management: Ensures the elevator operates efficiently and safely for all users.

Pre-condition(s):

The elevator system is operational.

The passenger is present in the building.

Success guarantee(s) (Post-conditions):

The passenger is safely transported to their desired floor.

The elevator doors operate correctly according to the passenger's input or system's default settings.

Main success scenario:

1. A passenger arrives at a specific floor.
2. The passenger presses the up or down button depending on their desired direction.
3. The elevator system acknowledges the call request.
4. The system identifies and dispatches the nearest available elevator to the passenger's floor.
5. The elevator arrives at the passenger's floor and the doors open.
6. The passenger enters the elevator.
7. Inside the elevator, the passenger selects their destination floor.
8. If needed, the passenger can use the “open door” or “close door” buttons to control the doors during the journey.
9. The elevator travels to the selected destination floor.
10. Upon arrival, the elevator doors open automatically.
11. The passenger exits the elevator.

Extensions:

3a. If a selected floor button is not working: The system prompts the passenger to choose another floor.

4a. If the “open door” button is held: The system keeps the door open until the button is released, or a safety timeout occurs.

4b. If the door encounters an obstacle while closing (Door Obstacle Detection): The system stops the door from closing and opens it again. If the obstruction persists, the system issues a warning message and audio sound.

5a. If a fire is detected (Fire Alarm Response): The system moves the elevator to a safe floor and instructs passengers to disembark while displaying a warning message and audio sound.

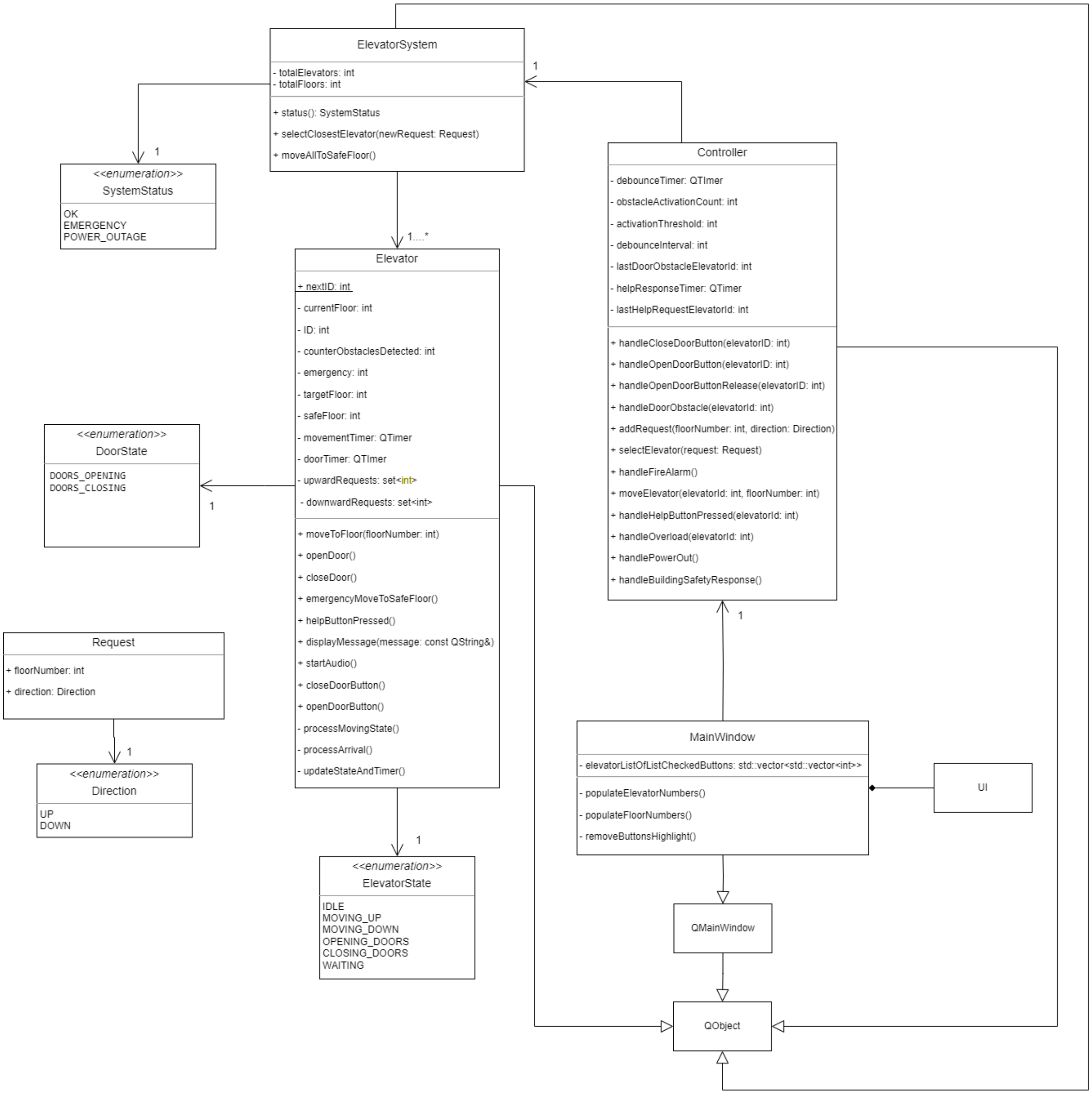
5b. If an overload is detected (Overload Detection): The system prevents the elevator from moving and alerts passengers to reduce the load.

5c. If a power outage occurs (Power Outage Handling): The system switches to backup power, moves the elevator to the nearest safe floor, and assists passengers in disembarking with a warning message and audio sound.

5d. If the Help button is pressed (Help Alarm): The system connects the passenger to Building Safety Service or places a 911 emergency call if no response is received.

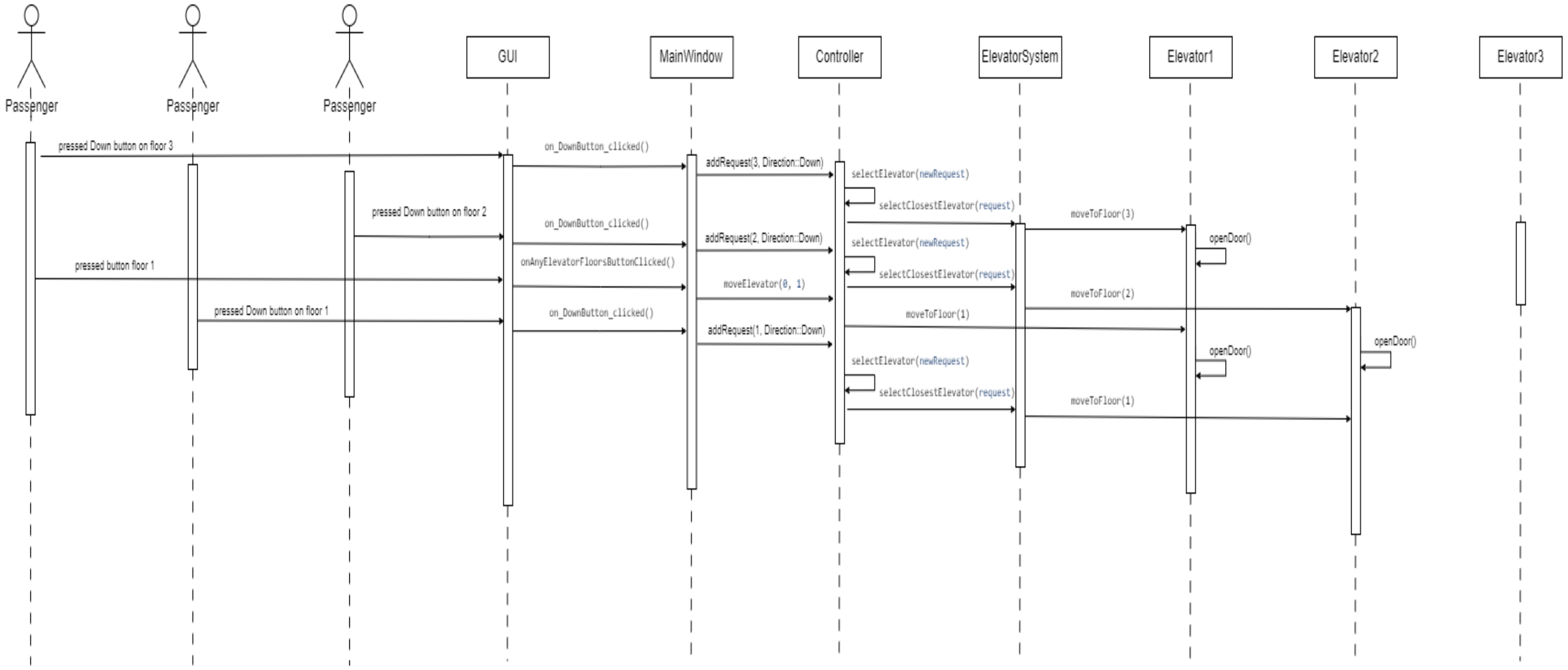
UML Class Diagram

Note: I got full marks for the UML in Assignment 2 but removed classes for physical objects (e.g., floor indicator, buttons, sensors) from the GUI. This is because they're redundant with a GUI, can slow the program, and don't align with real-life (where such objects are physical and interact with signals, not code). I'm handling these interactions with the QT framework, aiming for efficiency and realism.



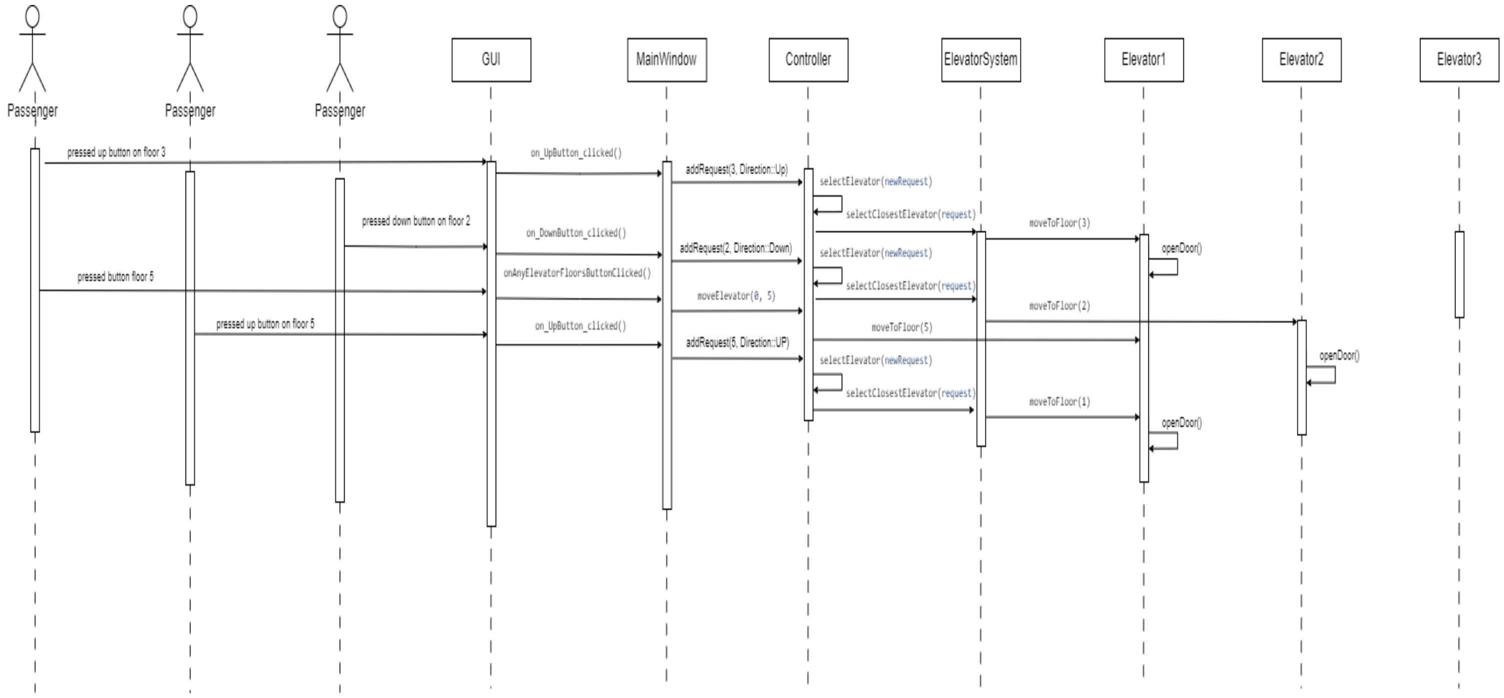
ElevatorSystem
- totalElevators: int
- totalFloors: int
+ status(): SystemStatus
+ selectClosestElevator(newRequest: Request)
+ moveAllToSafeFloor()
1
<<enumeration>>
SystemStatus
OK
EMERGENCY
POWER_OUTAGE
1
1....*
Controller
- debounceTimer: QTImer
- obstacleActivationCount: int
- activationThreshold: int
- debounceInterval: int
- lastDoorObstacleElevatorId: int
- helpResponseTimer: QTimer
- lastHelpRequestElevatorId: int
+ handleCloseDoorButton(elevatorID: int)
+ handleOpenDoorButton(elevatorID: int)
+ handleOpenDoorButtonRelease(elevatorID: int)
+ handleDoorObstacle(elevatorId: int)
+ addRequest(floorNumber: int, direction: Direction)
+ selectElevator(request: Request)
+ handleFireAlarm()
+ moveElevator(elevatorId: int, floorNumber: int)
+ handleHelpButtonPressed(elevatorId: int)
+ handleOverload(elevatorId: int)
+ handlePowerOut()
+ handleBuildingSafetyResponse()
Elevator
+ nextID: int
- currentFloor: int
- ID: int
- counterObstaclesDetected: int
- emergency: int
- targetFloor: int
- safeFloor: int
- movementTimer: QTimer
- doorTimer: QTimer
- upwardRequests: set<int>
- downwardRequests: set<int>
+ moveToFloor(floorNumber: int)
+ openDoor()
+ closeDoor()
+ emergencyMoveToSafeFloor()
+ helpButtonPressed()
+ displayMessage(message: const QString&)
+ startAudio()
+ closeDoorButton()
+ openDoorButton()
- processMovingState()
- processArrival()
- updateStateAndTimer()
<<enumeration>>
DoorState
DOORS_OPENING
DOORS_CLOSING
1
Request
+ floorNumber: int
+ direction: Direction
1
<<enumeration>>
Direction
UP
DOWN
1
<<enumeration>>
ElevatorState
IDLE
MOVING_UP
MOVING_DOWN
OPENING_DOORS
CLOSING_DOORS
WAITING
1
MainWindow
- elevatorListOfListCheckedButtons: std::vector<std::vector<int>>
- populateElevatorNumbers()
- populateFloorNumbers()
- removeButtonsHighlight()
UI
QMainWindow
QObject

Success Sequence Diagram 1

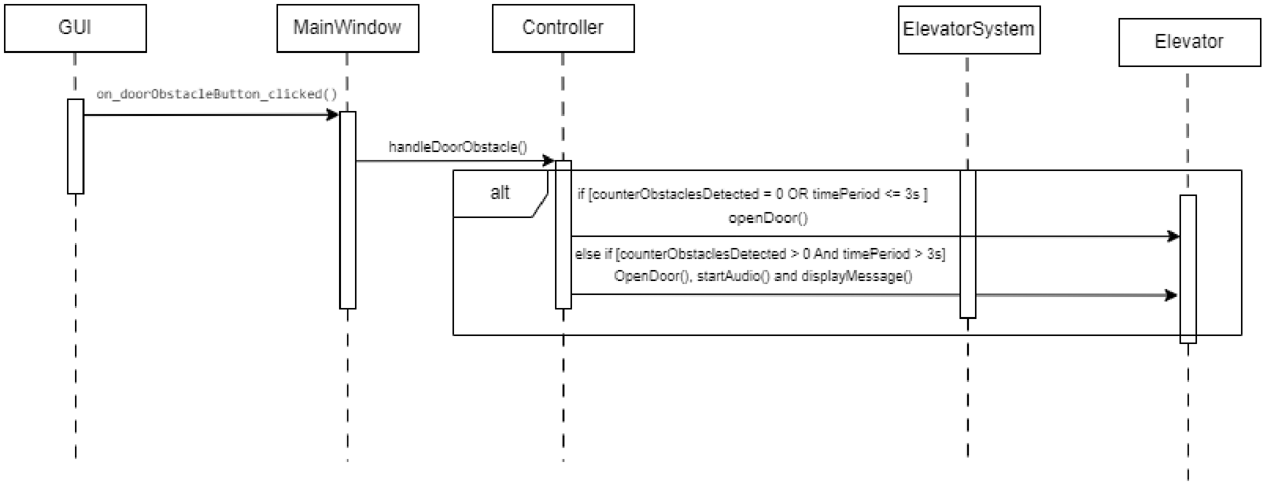

Passenger
Passenger
Passenger
GUI
MainWindow
Controller
ElevatorSystem
Elevator1
Elevator2
Elevator3
pressed Down button on floor 3
on_DownButton_clicked()
addRequest(3, Direction::Down)
selectElevator(newRequest)
selectClosestElevator(request)
moveToFloor(3)
pressed Down button on floor 2
on_DownButton_clicked()
addRequest(2, Direction::Down)
selectElevator(newRequest)
openDoor()
pressed button floor 1
onAnyElevatorFloorsButtonClicked()
selectClosestElevator(request)
moveElevator(0, 1)
moveToFloor(2)
pressed Down button on floor 1
on_DownButton_clicked()
moveToFloor(1)
addRequest(1, Direction::Down)
openDoor()
openDoor()
selectElevator(newRequest)
selectClosestElevator(request)
moveToFloor(1)

A TA told me it would be enough to just change the directions and the following outcomes

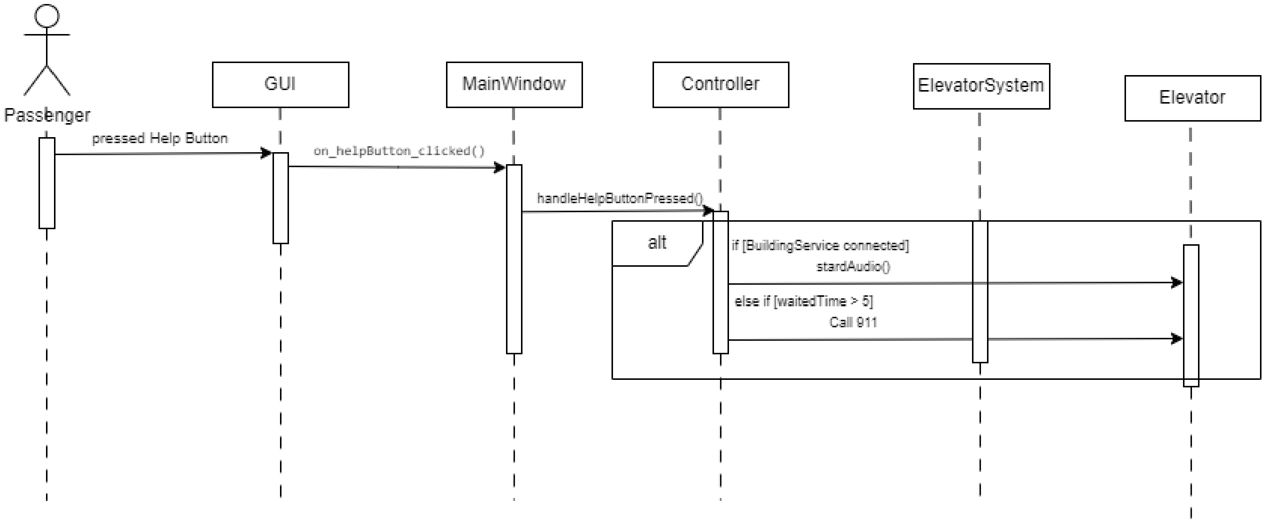
Success Sequence Diagram 2


Passenger
Passenger
Passenger
GUI
MainWindow
Controller
ElevatorSystem
Elevator1
Elevator2
Elevator3
pressed up button on floor 3
on_UpButton_clicked()
addRequest(3, Direction::Up)
selectElevator(newRequest)
selectClosestElevator(request)
pressed down button on floor 2
moveToFloor(3)
on_DownButton_clicked()
openDoor()
addRequest(2, Direction::Down)
selectElevator(newRequest)
pressed button floor 5
onAnyElevatorFloorsButtonClicked()
selectClosestElevator(request)
moveElevator(0, 5)
moveToFloor(2)
pressed up button on floor 5
on_UpButton_clicked()
moveToFloor(5)
addRequest(5, Direction::UP)
openDoor()
selectElevator(newRequest)
selectClosestElevator(request)
moveToFloor(1)
openDoor()

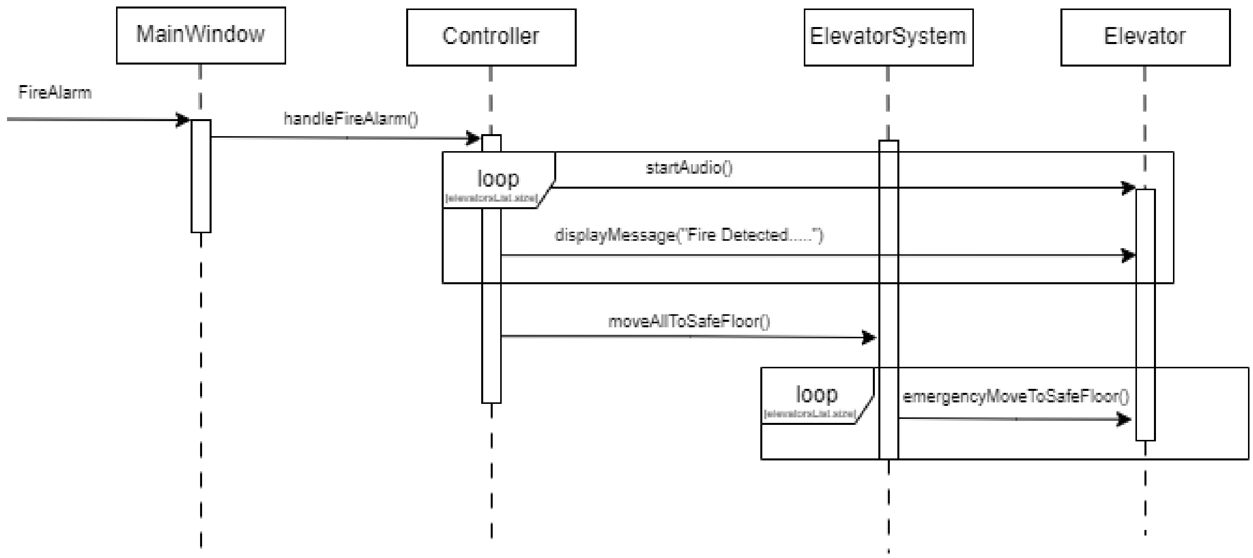
Door Obstacle Sequence Diagram


GUI
MainWindow
Controller
ElevatorSystem
Elevator
on_doorObstacleButton_clicked()
handleDoorObstacle()
alt
if [counterObstaclesDetected = 0 OR timePeriod <= 3s]
openDoor()
else if [counterObstaclesDetected > 0 And timePeriod > 3s]
OpenDoor(), startAudio() and displayMessage()

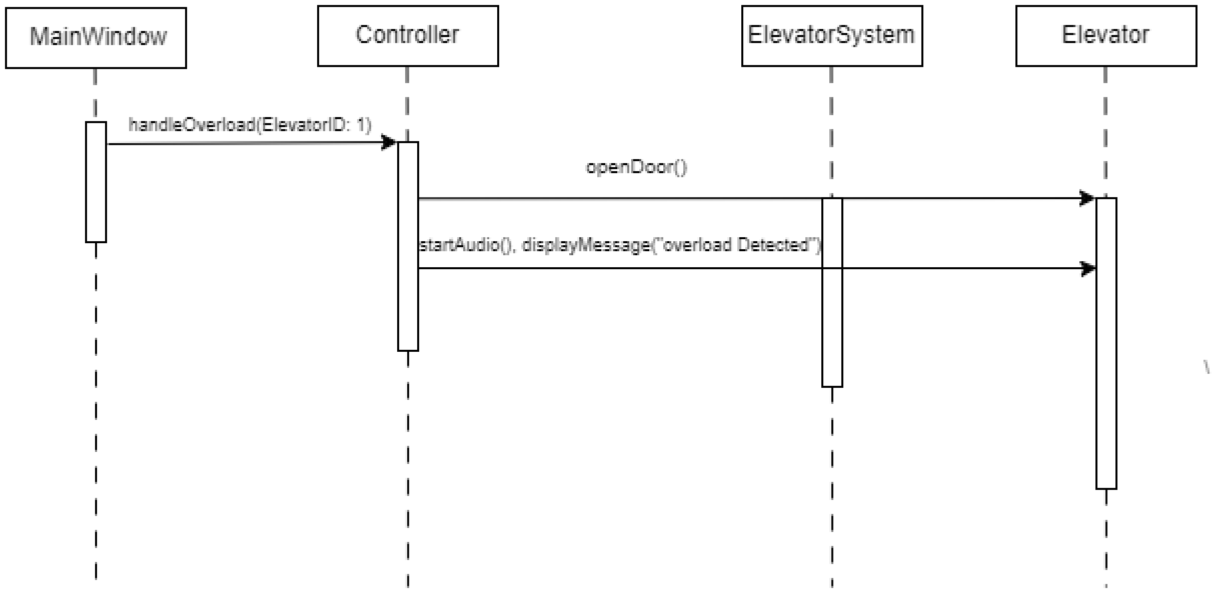
Help Button Sequence Diagram



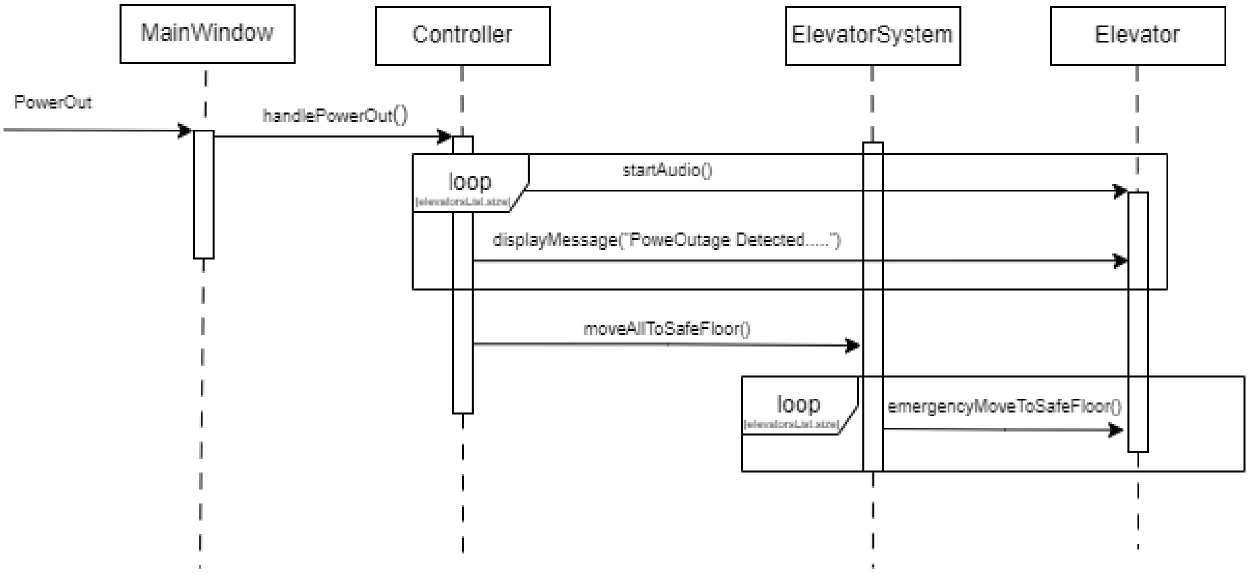
Fire Sequence Diagram



Overload Sequence Diagram

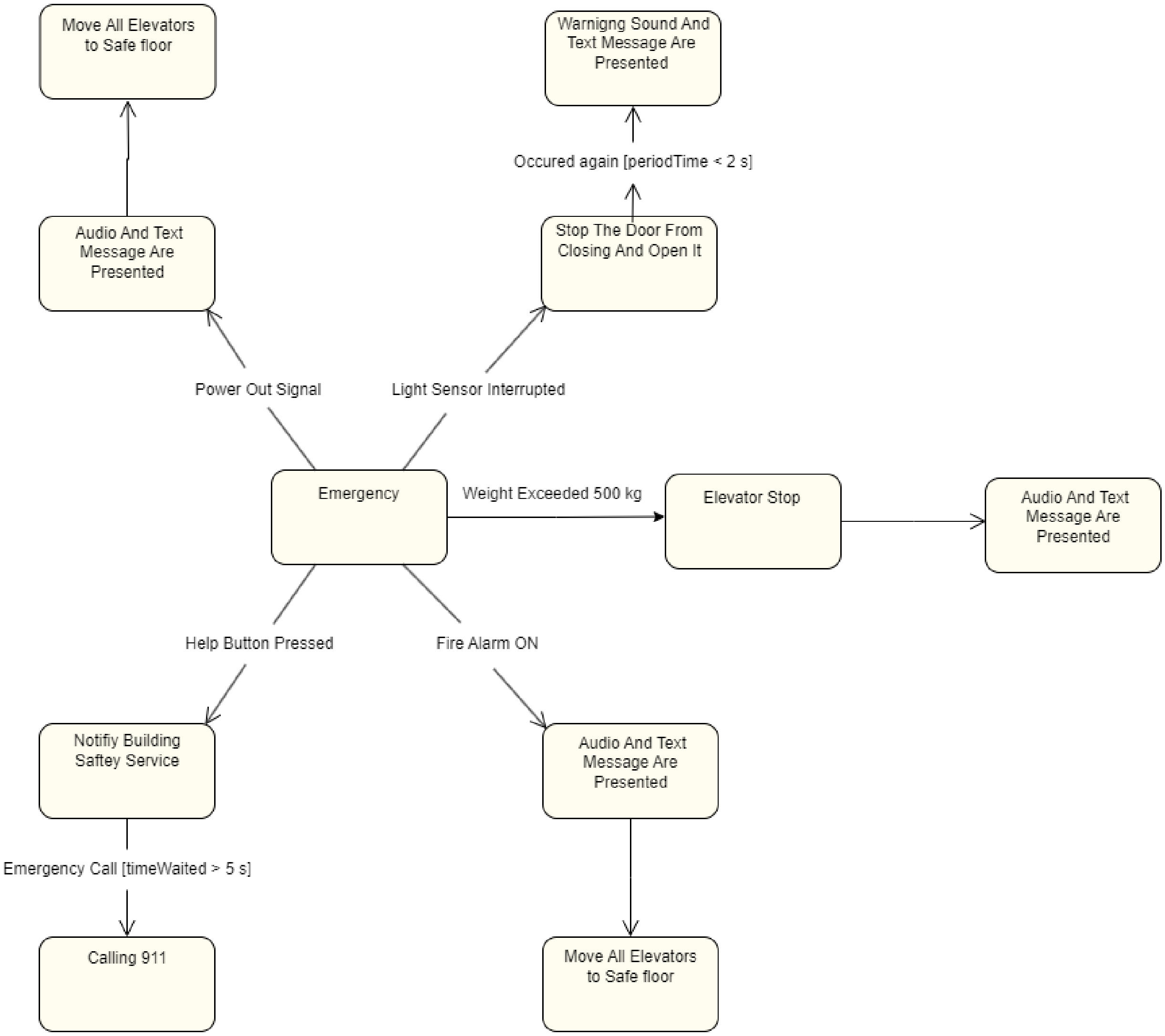

MainWindow
Controller
ElevatorSystem
Elevator
handleOverload(ElevatorID: 1)
openDoor()
startAudio(), displayMessage("overload Detected")

Power Out Sequence Diagram

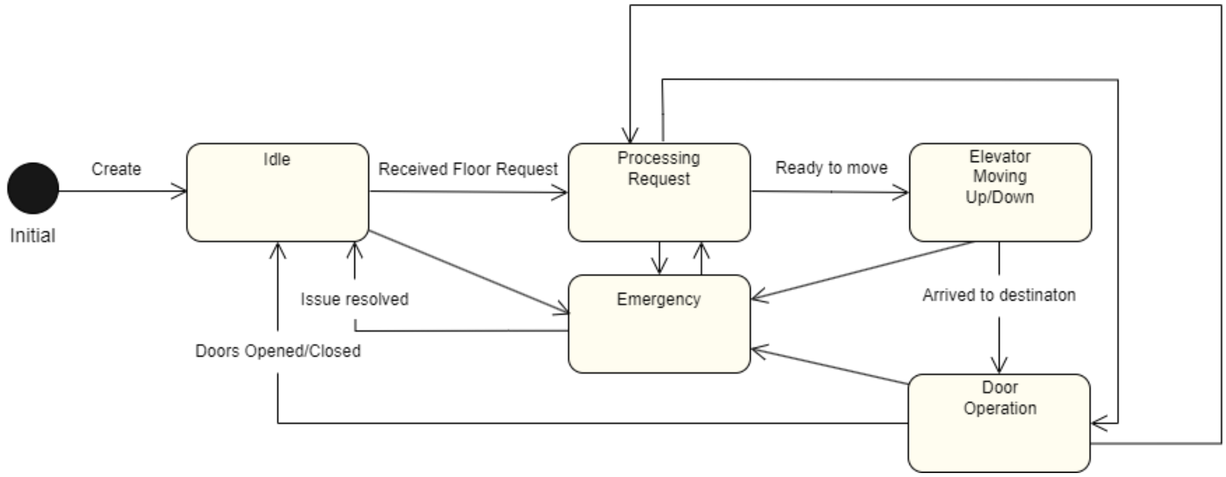


Emergency State Diagram

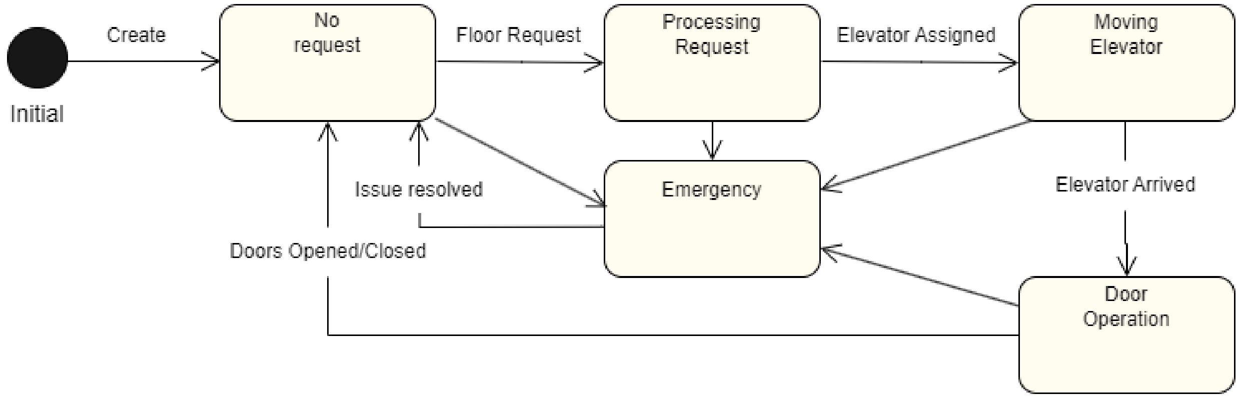
Note: this state diagram is done to explain the "Emergency" state in the 3 diagrams below


Move All Elevators to Safe floor
Warnigng Sound And Text Message Are Presented
Occured again [periodTime < 2 s]
Audio And Text Message Are Presented
Stop The Door From Closing And Open It
Power Out Signal
Light Sensor Interrupted
Emergency
Weight Exceeded 500 kg
Elevator Stop
Audio And Text Message Are Presented
Help Button Pressed
Fire Alarm ON
Notifiy Building Saftey Service
Audio And Text Message Are Presented
Emergency Call [timeWaited > 5 s]
Calling 911
Move All Elevators to Safe floor

Elevator State Diagram



Controller Sequence Diagram



Discussion

Design Decision:

In the centralized system, the Mediator pattern was applied by utilizing a central Controller class. This Controller acts as a mediator between the ElevatorSystem (which manages all elevators) and the MainWIndow (which receives user requests). The Controller processes incoming requests from different floors and then allocates these requests to the appropriate elevator based on current system status and logic.

The choice to use the Mediator pattern streamlines communication within the system by reducing the complexity of interactions between various components. Instead of having each elevator communicate directly with every floor panel, the centralized controller handles all communications and decisions. This setup simplifies the design and improves maintainability by centralizing request handling and elevator dispatching logic.

Traceability Matrix Diagram

ID	Requirement Description	Related Use Case	Fulfilled By (Centralized)	Implemented By	Tested By
1	Passenger can call an elevator from any floor by pressing the up or down button.	Steps 1, 2, 3	Controller ElevatorSystem Elevator MainWindow	MainWindow:: onAnyElevatorFloorsButtonClicked, Controller::addRequest	select a floor from the floorPanel ComboBox and press either the "up" or "down" button, verifying the system's response in dispatching the nearest elevator to the chosen floor.
2	System dispatches the nearest available elevator to the passenger's floor.	Steps 3, 4	Controller ElevatorSystem Elevator MainWindow	ElevatorSystem::selectClosestElevator	After pressing "up" or "down," check the console to confirm the nearest in-route elevator is dispatched to you.
3	Elevator arrives at the passenger's floor and doors open.	Steps 5, 6	Controller ElevatorSystem Elevator MainWindow	Elevator::moveToFloor, Elevator::openDoor	Check the console for confirmation that the dispatched elevator arrives and opens its door.
4	Passenger can select their destination floor inside the elevator.	Step 7	Controller ElevatorSystem Elevator MainWindow	MainWindow:: onAnyElevatorFloorsButtonClicked	Use the elevator panel to select the desired floor, ensuring to match the elevator with the passenger's floor using the ComboBox.
5	Passenger can control the elevator doors using the "open door" or "close door" buttons.	Step 8	Controller ElevatorSystem Elevator MainWindow	Elevator::openDoorButton, Elevator::closeDoorButton	Press the "open" or "close" button on the elevator panel and verify through the console if the elevator door opens and remains stationary or closes as intended.

6	Elevator moves to the selected destination floor.	Step 9	Controller ElevatorSystem Elevator MainWindow	Elevator::moveToFloor	Check the console and track the display to confirm the elevator door's behavior.
7	Elevator doors open automatically upon arriving at the destination floor.	Step 10	Elevator	Elevator::openDoor within Elevator::updateFloor	Check the console and track the display to confirm the elevator door's behavior.
8	Passenger exits the elevator at their destination floor.	Step 11	NA	NA	The door stays open for 10 seconds to ensure the passenger gets out safely. Check the console
9	System provides help through a connection to building safety service when Help button is pressed.	Safety Feature: Help	Controller ElevatorSystem Elevator MainWindow	MainWindow::on_helpButton_clicked , Controller::handleHelpButtonPressed	Click the red help button and observe the console to note the "waiting for building service response." To respond before 5 seconds elapse, press the response button; otherwise, check the console for the "call 911" message
10	System reopens doors if an obstacle is detected during closing.	Safety Feature: Door Obstacles	Controller ElevatorSystem Elevator MainWindow	Controller::handleDoorObstacle, Elevator::openDoor	Click the door obstacle button and observe the console. For a specific message to display, click the button multiple times in succession.
11	System redirects all elevators to a safe floor in case of fire alarm.	Safety Feature: Fire	Controller ElevatorSystem Elevator MainWindow	Controller::handleFireAlarm, ElevatorSystem::moveAllTOSafeFloor	Click the fire button and review the console to confirm a message is sent to every elevator, directing all elevators to proceed to the safe floor.

12	Elevator prevents movement and alerts passengers if an overload is detected.	Safety Feature: Overload	Controller ElevatorSystem Elevator MainWindow	MainWindow:: on_WeightTestButton_clicked, Controller::handleOverload	Enter a number in the weight text edit and click test; if it's above 500, a message will be sent to the elevator, which can be verified in the console.
13	System ensures safe disembarkation of passengers in case of power outage.	Safety Feature: Power Out	Controller ElevatorSystem Elevator MainWindow	Controller::handlePowerOut, ElevatorSystem::moveAllTOSafeFloor	Click the power outage button and observe the console to see a message sent to every elevator, indicating that all elevators are moving to the safe floor.